has only 2^{b-1} bits available. Explain how your colleague can use your Bloom filter to avoid rebuilding a new Bloom filter using the original dictionary of words.

Exercise 5.25: For the leader election problem alluded to in Section 5.5.4, we have n users, each with an identifier. The hash function takes as input the identifier and outputs a b-bit hash value, and we assume that these values are independent and uniformly distributed. Each user hashes its identifier, and the leader is the user with the smallest hash value. Give lower and upper bounds on the number of bits b necessary to ensure that a unique leader is successfully chosen with probability p. Make your bounds as tight as possible.

Exercise 5.26: Consider Algorithm 5.2, the modified algorithm for finding Hamiltonian cycles. We have shown that the algorithm can be applied to find a Hamiltonian cycle with high probability in a graph chosen randomly from $G_{n,p}$, when p is known and sufficiently large, by initially placing edges in the edge lists appropriately. Argue that the algorithm can similarly be applied to find a Hamiltonian cycle with high probability on a graph chosen randomly from $G_{n,N}$ when $N = c_1 n \ln n$ for a suitably large constant c_1. Argue also that the modified algorithm can be applied even when p is not known in advance as long as p is at least $c_2 \ln n/n$ for a suitably large constant c_2.

5.8. An Exploratory Assignment

Part of the research process in random processes is first to understand what is going on at a high level and then to use this understanding in order to develop formal mathematical proofs. In this assignment, you will be given several variations on a basic random process. To gain insight, you should perform experiments based on writing code to simulate the processes. (The code should be very short, a few pages at most.) After the experiments, you should use the results of the simulations to guide you to make conjectures and prove statements about the processes. You can apply what you have learned up to this point, including probabilistic bounds and analysis of balls-and-bins problems.

Consider a complete binary tree with $N = 2^n - 1$ nodes. Here n is the depth of the tree. Initially, all nodes are *unmarked*. Over time, via processes that we shall describe, nodes becomes *marked*.

All of the processes share the same basic form. We can think of the nodes as having unique identifying numbers in the range of $[1, N]$. Each unit of time, I *send* you the

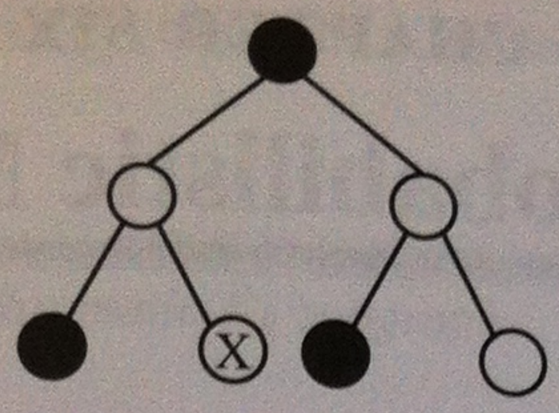

Figure 5.3: The arrival of X causes all other nodes to be marked.

identifier of a node. When you receive a sent node, you mark it. Also, you invoke the following *marking rule*, which takes effect before I send out the next node.

- If a node and its sibling are marked, its parent is marked.
- If a node and its parent are marked, the other sibling is marked.

The marking rule is applied *recursively as much as possible* before the next node is sent. For example, in Figure 5.3, the marked nodes are filled in. The arrival of the node labeled by an X will allow you to mark the remainder of the nodes, as you apply the marking rule first up and then down the tree. Keep in mind that you always apply the marking rule as much as possible.

Now let us consider the different ways in which I might be sending you the nodes.

Process 1: Each unit of time, I send the identifier of a node chosen *independently and uniformly at random from all of the N nodes.* Note that I might send you a node that is already marked, and in fact I may send a useless node that I have already sent.

Process 2: Each unit of time I send the identifier of a node chosen uniformly at random from those nodes *that I have not yet sent.* Again, a node that has already been marked might arrive, but each node will be sent at most once.

Process 3: Each unit of time I send the identifier of a node chosen uniformly at random from those nodes *that you have not yet marked.*

We want to determine how many time steps are needed before all the nodes are marked for each of these processes. Begin by writing programs to simulate the sending processes and the marking rule. Run each process ten times for each value of n in the range [10, 20]. Present the data from your experiments in a clear, easy-to-read fashion and explain your data suitably. A tip: You may find it useful to have your program print out the last node that was sent before the tree became completely marked.

1. For the first process, prove that the expected number of nodes sent is $\Omega(N \log N)$. How well does this match your simulations?
2. For the second process, you should find that almost all N nodes must be sent before the tree is marked. Show that, with constant probability, at least $N - 2\sqrt{N}$ nodes must be sent.
3. The behavior of the third process might seem a bit unusual. Explain it with a proof.

After answering these questions, you may wish to consider other facts you could prove about these processes.